Contents

Classifying Metrics Based on Request or Response

6 minute read page test

It’s useful to visualize telemetry based on the type of requests and responses handled by services in your mesh. For example, a bookseller tracks the number of times book reviews are requested. A book review request has this structure:

```
GET /reviews/{review_id}
```

Counting the number of review requests must account for the unbounded element `review_id`. `GET /reviews/1` followed by `GET /reviews/2` should count as two requests to get reviews.

Istio lets you create classification rules using the AttributeGen plugin that groups requests into a fixed number of logical operations. For example, you can create an operation named `GetReviews`, which is a common way to identify operations using the `Open API Spec operationId`. This information is injected into request processing as `istio_operationId` attribute with value equal to `GetReviews`. You can use the attribute as a dimension in Istio standard metrics. Similarly, you can track metrics based on other operations like

`ListReviews` and `CreateReviews`.

For more information, see the reference content.

Istio uses the Envoy proxy to generate metrics and provides its configuration in the `EnvoyFilter` at manifests/charts/istio-control/istio-discovery/templates/telemetryv2_1.11.yaml. As a result, writing classification rules involves adding attributes to the `EnvoyFilter`.

Classify metrics by request

You can classify requests based on their type, for example `ListReview`, `GetReview`, `CreateReview`.

1. Create a file, for example `attribute_gen_service.yaml`, and save it with the following contents. This adds the `istio.attributegen` plugin to the `EnvoyFilter`. It also creates an attribute, `istio_operationId` and populates it with values for the categories to count as metrics.

   This configuration is service-specific since request paths are typically service-specific.

   ```
   apiVersion: networking.istio.io/v1alpha3
   kind: EnvoyFilter
   metadata:
     name: istio-attributegen-filter
   spec:
     workloadSelector:
       labels:
         app: reviews
     configPatches:
     - applyTo: HTTP_FILTER
   ```

```
    match:
      context: SIDECAR_INBOUND
      proxy:
        proxyVersion: '1\.9.*'
      listener:
        filterChain:
          filter:
            name: "envoy.http_connection_manage
r"
            subFilter:
              name: "istio.stats"
    patch:
      operation: INSERT_BEFORE
      value:
        name: istio.attributegen
        typed_config:
          "@type": type.googleapis.com/udpa.typ
e.v1.TypedStruct
          type_url: type.googleapis.com/envoy.e
xtensions.filters.http.wasm.v3.Wasm
          value:
            config:
              configuration:
                "@type": type.googleapis.com/go
ogle.protobuf.StringValue
                value: |
                  {
                    "attributes": [
                      {
                        "output_attribute": "is
tio_operationId",
                        "match": [
                          {
                            "value": "ListRevie
ws",
                            "condition": "reque
st.url_path == '/reviews' && request.method ==
'GET'"
                          },
                          {
                            "value": "GetReview
",
                            "condition": "reque
st.url_path.matches('^/reviews/[[:alnum:]]*$')
&& request.method == 'GET'"
                          },
                          {
                            "value": "CreateRev
iew",
                            "condition": "reque
st.url_path == '/reviews/' && request.method ==
 'POST'"
                          }
                        ]
                      }
                    ]
                  }
              vm_config:
                runtime: envoy.wasm.runtime.nul
l
                code:
                  local: { inline_string: "envo
```

```
y.wasm.attributegen" }
```

2. Apply your changes using the following command:

   ```
   $ kubectl -n istio-system apply -f attribute_ge
   n_service.yaml
   ```

3. Find the `stats-filter-1.11` `EnvoyFilter` resource from the `istio-system` namespace, using the following command:

   ```
   $ kubectl -n istio-system get envoyfilter | gre
   p ^stats-filter-1.11
   stats-filter-1.11                    2d
   ```

4. Create a local file system copy of the `EnvoyFilter` configuration, using the following command:

   ```
   $ kubectl -n istio-system get envoyfilter stats
   -filter-1.11 -o yaml > stats-filter-1.11.yaml
   ```

5. Open `stats-filter-1.11.yaml` with a text

editor and locate the `name: istio.stats` extension configuration. Update it to map `request_operation` dimension in the `requests_total` standard metric to `istio_operationId` attribute. The updated configuration file section should look like the following.

```
name: istio.stats
typed_config:
  '@type': type.googleapis.com/udpa.type.v1.Typ
edStruct
  type_url: type.googleapis.com/envoy.extension
s.filters.http.wasm.v3.Wasm
  value:
    config:
      configuration:
        "@type": type.googleapis.com/google.pro
tobuf.StringValue
        value: |
          {
            "metrics": [
             {
               "name": "requests_total",
               "dimensions": {
                 "request_operation": "istio_op
erationId"
               }
             }]
          }
```

6. Save `stats-filter-1.11.yaml` and then apply the configuration using the following command:

```
$ kubectl -n istio-system apply -f stats-filter
-1.11.yaml
```

7. Add the following configuration to the mesh config. This results in the addition of the `request_operation` as a new dimension to the `istio_requests_total` metric. Without it, a new metric with the name `envoy_request_operation___somevalue___istio_requests_total` is created.

   ```
   meshConfig:
     defaultConfig:
       extraStatTags:
       - request_operation
   ```

8. Generate metrics by sending traffic to your application.
9. After the changes take effect, visit Prometheus and look for the new or changed dimensions, for example `istio_requests_total`.

Classify metrics by response

You can classify responses using a similar process as requests. Do note that the `response_code` dimension already exists by default. The example below will change how it is populated.

1. Create a file, for example `attribute_gen_service.yaml`, and save it with the following contents. This adds the `istio.attributegen` plugin to the `EnvoyFilter` and generates the `istio_responseClass` attribute used by the stats plugin.

   This example classifies various responses, such as grouping all response codes in the `200` range as a `2xx` dimension.

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: istio-attributegen-filter
spec:
  workloadSelector:
    labels:
      app: productpage
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      context: SIDECAR_INBOUND
      proxy:
        proxyVersion: '1\.9.*'
      listener:
        filterChain:
          filter:
            name: "envoy.http_connection_manage
r"
            subFilter:
              name: "istio.stats"
    patch:
      operation: INSERT_BEFORE
      value:
        name: istio.attributegen
        typed_config:
          "@type": type.googleapis.com/udpa.typ
e.v1.TypedStruct
          type_url: type.googleapis.com/envoy.e
```

```
xtensions.filters.http.wasm.v3.Wasm
          value:
            config:
              configuration:
                "@type": type.googleapis.com/go
ogle.protobuf.StringValue
                value: |
                  {
                    "attributes": [
                      {
                        "output_attribute": "is
tio_responseClass",
                        "match": [
                          {
                            "value": "2xx",
                            "condition": "respo
nse.code >= 200 && response.code <= 299"
                          },
                          {
                            "value": "3xx",
                            "condition": "respo
nse.code >= 300 && response.code <= 399"
                          },
                          {
                            "value": "404",
                            "condition": "respo
nse.code == 404"
                          },
                          {
                            "value": "429",
                            "condition": "respo
nse.code == 429"
                          },
                          {
                            "value": "503",
                            "condition": "respo
nse.code == 503"
                          },
                          {
                            "value": "5xx",
                            "condition": "respo
nse.code >= 500 && response.code <= 599"
                          },
                          {
                            "value": "4xx",
                            "condition": "respo
nse.code >= 400 && response.code <= 499"
                          }
                        ]
                      }
                    ]
                  }
              vm_config:
                runtime: envoy.wasm.runtime.nul
l
                code:
                  local: { inline_string: "envo
y.wasm.attributegen" }
```

2. Apply your changes using the following command:

```
$ kubectl -n istio-system apply -f attribute_ge
n_service.yaml
```

3. Find the `stats-filter-1.11` `EnvoyFilter` resource from the `istio-system` namespace, using the following command:

   ```
   $ kubectl -n istio-system get envoyfilter | gre
   p ^stats-filter-1.11
   stats-filter-1.11                   2d
   ```

4. Create a local file system copy of the `EnvoyFilter` configuration, using the following command:

   ```
   $ kubectl -n istio-system get envoyfilter stats
   -filter-1.11 -o yaml > stats-filter-1.11.yaml
   ```

5. Open `stats-filter-1.11.yaml` with a text editor and locate the `name: istio.stats` extension configuration. Update it to map `response_code` dimension in the `requests_total` standard metric to

`istio_responseClass` attribute. The updated configuration file section should look like the following.

```
name: istio.stats
typed_config:
  '@type': type.googleapis.com/udpa.type.v1.Typ
edStruct
  type_url: type.googleapis.com/envoy.extension
s.filters.http.wasm.v3.Wasm
  value:
    config:
      configuration:
        "@type": type.googleapis.com/google.pro
tobuf.StringValue
        value: |
          {
            "metrics": [
             {
               "name": "requests_total",
               "dimensions": {
                 "response_code": "istio_respon
seClass"
               }
             }]
          }
```

6. Save `stats-filter-1.11.yaml` and then

apply the configuration using the following command:

```
$ kubectl -n istio-system apply -f stats-filter
-1.11.yaml
```

Verify the results

1. Generate metrics by sending traffic to your application.
2. Visit Prometheus and look for the new or changed dimensions, for example `2xx`. Alternatively, use the following command to verify that Istio generates the data for your new dimension:

   ```
   $ kubectl exec pod-name -c istio-proxy -- curl
   -sS 'localhost:15000/stats/prometheus' | grep i
   stio_
   ```

In the output, locate the metric (e.g. `istio_requests_total`) and verify the presence of the new or changed dimension.

Troubleshooting

If classification does not occur as expected, check the following potential causes and resolutions.

Review the Envoy proxy logs for the pod that has the service on which you applied the configuration change. Check that there are no errors reported by the service in the Envoy proxy logs on the pod, (`pod-name`), where you configured classification by using the following command:

```
$ kubectl logs pod-name -c istio-proxy | grep -e "C
onfig Error" -e "envoy wasm"
```

Additionally, ensure that there are no Envoy proxy crashes by looking for signs of restarts in the output of the following command:

```
$ kubectl get pods pod-name
```